Machine Learning in Penetration Testing

Huynh Nguyen Uyen Nhi, Tran Minh Duy, Pham Ngoc Thien

1 GIỚI THIỆU

Penetration testing (hay Pentest) là các hoạt động bảo mật nhằm đánh giá độ an toàn của hệ thống bằng cách mô phỏng lại các cuộc tấn công lên hệ thống. Việc pentest thường được thực hiện bởi kẻ tấn công là những người đã được đào tạo và phụ thuộc nhiều vào tính chuyên môn. Việc tự động hóa các hoạt động pentest là công việc khó khăn vì phạm vi công việc mà một chuyên gia con người có thể thực hiện lẫn kiến thức của người đó để đưa ra quyết định là không thể biết trước. Trong bài báo này chúng tôi tập trung đơn giản hóa các vấn đề pentest trên cơ sở các bài tập Capture The Flag (CTF) và chúng tôi sẽ phân tích cách các thuật toán model-free reinforcement learning giúp giải quyết các vấn đề này. Trong khi mô hình hóa các bài tập CTF dưới dạng reinforcement learning chúng tôi nhấn mạnh rằng một bài tập cụ thể mô tả vấn đề pentest là việc tìm cấu trúc của vấn đề bằng cách thủ công. Sau đó chúng tôi đơn giản hóa công việc bằng việc dựa trên các kiến thức trước đó đã được cung cấp. Bằng cách này chúng tôi trình bày cách pentest được thực hiện dựa trên việc sử dụng phối hợp các thuật toán model-free và model-based. Sử dụng kỹ thuật này nhằm thêm vào các kiến thức cho trước, chúng tôi chỉ ra rằng việc này giúp định hướng agent tốt hơn và thu hẹp phạm vi phân tích vấn đề, từ đó thu được giải pháp hiệu quả hơn.

An toàn hệ thống và hạ tầng hiện đại là vấn đề trung tâm của bảo mật máy tính. Cùng với sự gia tăng của việc truyền dữ liệu qua nền tảng điện tử thì việc đảm bảo sự vận hành chính xác của chúng là rất quan trọng trong xã hội hiện đại. Phương pháp tiếp cận truyền thống để đánh giá độ an toàn của hệ thống là phân tích và củng cố phòng thủ của hệ thống dưới góc nhìn bảo vệ hệ thống. Một phương pháp tiếp cận khác chủ động hơn là penetration testing (pentest, PT) hay ethical hacking, bao gồm việc mô phỏng các tấn công mạng lên hệ thống thông tin với mục tiêu tìm các điểm yếu và đánh giá an toàn chung. Nhiều phần mềm được phát triển nhằm thực hiện một số khía cạnh của pentest, tuy nhiên chúng vẫn cần sự điều hướng của người dùng để thực hiện công việc của người pentest. Phương pháp truyền thống là sử dụng trí tuệ nhân tạo như là *planning* hướng tới việc tự động hóa pentest dựa trên việc tạo các kế hoạch pentest. Tuy nhiên đầu vào từ con người phụ thuộc rất nhiều từ ngữ cảnh và hệ thống mục tiêu, cũng như kết luận về sự xuất hiện của lỗ hổng.

Các tiến bộ gần đây đối với trí tuệ nhân tạo và máy học cho phép cải thiện một số hạn chế của việc tự động hóa PT. Cụ thể, reinforcement learning (RL) được chứng minh là phương pháp hiệu quả nhằm giải quyết các vấn đề phức tạp mà người sử dụng cố gắng tối ưu lời giải trên những môi trường cho trước. RL ứng dụng rất nhiều thuật toán với các mức độ tính toán và độ phức tạp của mẫu khác nhau, cụ thể thì thuật toán model-free dựa trên lượng nhỏ kiến thức có trước đó của một môi trường, và điều này cho phép đào tạo agent chỉ bởi việc tương tác với môi trường, tương tự việc người chơi tương tác với game để tìm ra hướng giải quyết. Game cung cấp benchmark trong một khoảng thời gian dài cho RL, và các phương pháp model-free đã đạt được các kết quả mới nhất nhằm giải quyết các trò chơi phức tạp, từ cờ vây truyền thống đến Atari hiện đại. Sự phát triển này cho thấy khả năng ứng dụng model-free RL vào việc giải quyết các vấn đề PT. CTF, như một dạng trò chơi của PT, cung cấp khả năng deploy các thuật toán RL và đào tạo agent, cho phép học và hoàn thành PT độc lập với giám sát của con người.

Trong bài viết này, chúng tôi giải quyết câu hỏi về mức độ mà thuật toán RL model-free có thể được sử dụng để giải quyết các thử thách CTF. Mặc dù việc áp dụng RL model-free để giải quyết các thử thách CTF có vẻ hoàn toàn phù hợp nhưng chúng tôi nhấn mạnh tầm quan trọng cụ thể vốn có trong PT: tính tối nghĩa là sự khó khăn trong việc xác định cấu trúc tiềm ẩn của thử thách CTF. Điều này có thể là do hệ thống CTF ngăn chặn bất kỳ loại rò rỉ thông tin nào hoặc do sự hiện diện của các cơ chế bảo vệ giúp hệ thống mục tiêu điều chỉnh theo hành động của agent. Chúng tôi phân tích những vấn đề này bằng thực nghiệm, xem xét các kịch bản với entropy cao, đồng thời đánh giá các kỹ thuật RL khác nhau như *lazy loading*, *state aggression* và *imitation learning* có thể giúp giải quyết những thử thách này như thế nào.

Cuối cùng, chúng tôi chỉ ra rằng, mặc dù về nguyên tắc RL có thể cho phép học không cần mô hình (model-free), nhưng trên thực tế, việc dựa vào một số dạng kiến thức có sẵn có thể cần thiết để giải quyết được vấn đề. Chúng tôi lập luận rằng RL cung cấp một hướng nghiên cứu thú vị cho PT không phải vì nó cho phép học thuần túy không mô hình (ngược lại với các thuật toán trí tuệ nhân tạo dựa trên mô hình truyền thống), mà bởi vì nó có thể cung cấp một cách linh hoạt hơn để đánh đổi lượng kiến thức trước đó mà một agent được cung cấp và lượng cấu trúc mà một agent dự kiến sẽ tìm hiểu. Chúng tôi lập luận rằng việc đánh giá sự đánh đổi này và agent tận dụng được điều này sẽ tạo thành một hướng đi hiệu quả.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 CƠ SỞ LÝ THUYẾT

Trong phần này chúng tôi trình bày các ý tưởng cơ bản của PT và RL và xem xét các ứng dụng máy học vào các vấn đề PT.

2.1 Penetration Testing

Các hệ thống máy tính hiện đại có thể chứa các lỗ hổng ở nhiều dạng, từ các lỗ hổng phần mềm low-level tới các lỗ hổng về dịch vụ mạng. Các lỗ hổng này có thể được nhắm tới bởi kẻ tấn công bằng nhiều cách thức lẫn động lực. PT dựa trên các khía cạnh đó của hacker nhằm thực hiện các tấn công và tìm ra các lỗ hổng tiềm tàng trong hệ thống.

2.1.1 Hacking attacks. Chúng ta không có một quy luật chung cho việc triển khai tấn công mạng nhưng có thể chỉ ra một số bước thường thấy khi thực hiện tấn công trong nhiều trường hợp. Trên góc độ kẻ tấn công thì hacking gồm bước *thu thập thông tin* và bước *khai thác lỗ hổng*.

Trong giai đoạn thứ nhất kẻ tấn công thu thập các thông tin trên hệ thống mục tiêu (ví dụ như tìm trên website các thông tin có ích, xác định input ở server side, ...). Quá trình này cho phép xác định sự liên quan của thông tin và sự hiện diện của các lỗ hổng thường gây ra "nghẽn cổ chai" trong quá trình tấn công. Các lỗ hổng khác nhau và giai đoạn thứ hai về khai thác lỗ hổng yêu cầu kiến thức về hệ thống mục tiêu cũng như các hành động để xác định điểm yếu. Kẻ tấn công thường dựa trên rất nhiều năng lực, từ logic đến trực giác, từ kiến thức chuyên môn đến kinh nghiệm có trước đó. Sau khi khai thác thành công kẻ tấn công cũng có thể thực hiện theo nhiều cách khác nhau dựa trên mục tiêu tấn công. Đó có thể là giữ một kênh không chính thống luôn mở nhằm trích xuất thông tin bí mật hoặc dùng để thực hiện nhiều tấn công khác.

Một ví dụ điển hình của PT là web hacking. Quá trình web hacking có thể được triển khai theo nhiều cách khác nhau với các bước khác nhau. Cụ thể, kẻ tấn công bắt đầu bằng việc xác định dịch vụ web thông qua port scanning và bằng việc thiết lập liên lạc với dịch vụ thông qua giao thức HTTP. Sau đó kẻ tấn công có thể truy cập tới thư mục webroot, tải chúng xuống, xử lý trên trình duyệt và thực thi client-side script ở local. Dữ liệu có thể được xử lý trên file remote để chúng được xử lý trên server-side và kẻ tấn công phân tích web responses. Các script server-side có thể thực hiện nhiều hành động phức tạp như là truy vấn cơ sở dữ liệu, đọc và ghi lên local files, từ đó kẻ tấn công có thể gửi các input phù hợp để biết được các hành động trên server-side. Mặc dù website có thể chứa các lỗ hổng khác nhau hoặc lỗ hổng duy nhất, các lỗ hổng thường gặp có thể được xác định và phân loại.

2.1.2 Các cuộc thi hacking Capture The Flag. CTF là một nền tảng học tập thực tế dành cho ethical hacking.

Các sự kiện CTF thường được tổ chức dưới dạng các cuộc thi kéo dài 48 giờ, trong đó những thử thách bảo mật khác nhau được đưa ra cho người tham gia giải quyết.

Các cuộc thi CTF thường đưa ra một loạt các thử thách dựa trên các trường hợp bảo mật thực tế. Mỗi thử thách được xác định bởi một lỗ hổng (hoặc một chuỗi lỗ hổng) được liên kết với một cờ (flag). Mục đích của người tham gia là khai thác lỗ hổng trong mỗi thử thách và từ đó chiếm được cờ liên quan. Người chơi không cần thực hiện thêm bước nào khác (ví dụ như gửi dữ liệu đến máy chủ cuộc thi và điều khiển hoặc duy trì quyền truy cập); Việc lấy được một lá cờ tương đương việc một thử thách đã được giải quyết. Các thử thách có thể được phân loại theo loại vấn đề mà chúng đưa ra (ví dụ như thử thách web hacking hoặc khai thác mã nhị phân). Thông thường, yếu tố con người không can thiệp vào cách giải quyết, do đó kẻ tấn công phải dựa vào kiến thức và lý luận của họ chứ không phải dựa vào social engineering. Trong một số trường hợp, thông tin về hệ thống đích và lỗ hổng bảo mật có thể được cung cấp cho người tham gia.

CTF tiêu chuẩn ở chế độ Jeopardy, nghĩa là tất cả những người tham gia đều là những kẻ tấn công và họ phải đối mặt với một loạt các thử thách tĩnh khác nhau. Trong các biến thể khác, những người tham gia có thể được chia thành đội đỏ (red team), tức là đội tập trung vào tấn công hệ thống mục tiêu và đội xanh (blue team), là đội có nhiệm vụ bảo vệ hệ thống mục tiêu. Ngoài ra, mỗi đội có thể được cung cấp cơ sở hạ tầng mà họ phải bảo vệ đồng thời tấn công cơ sở hạ tầng của các đội khác. Hai biến thể cuối cùng của CTF này xác định các lỗ hổng không cố định, đang phát triển, vì những người bảo vệ trong đội xanh có thể thay đổi dịch vụ trong thời gian chạy bằng cách quan sát hành động của đội đỏ và vá các lỗ hổng của chính họ.

Tóm lại, CTF, đặc biệt là trong chế độ Jeopardy, xác định một tập hợp các vấn đề được xác định có thể nắm bắt được bản chất của PT và có thể dễ dàng áp dụng vào trò chơi.

2.2 Reinforcement Learning

Mô hình học tăng cường (Reinforcement Learning, RL) cung cấp một phương án linh hoạt để mô hình hóa các vấn đề phức tạp và giải quyết chúng bằng các thuật toán học tập có mục đích chung. Vấn đề RL thể hiện vấn đề của một agent đang cố gắng học một hành vi tối ưu trong một môi trường nhất định. Trong học tập không có mô hình, agent được cung cấp thông tin tối thiểu về môi trường, động lực của nó cũng như bản chất hoặc tác động của các hành động mà nó có thể thực hiện; thay vào đó, agent dự kiến sẽ học một hành vi hợp lý bằng cách tương tác với môi trường, từ đó khám phá hành động nào ở trạng thái nào có lợi hơn và cuối cùng xác định phương án cho phép nó đạt được mục tiêu của mình theo cách tốt nhất có thể.

2.2.1 Định nghĩa vấn đề RL. Một vấn đề RL được định nghĩa bởi một *tuple* hoặc *signature*:

$$\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$$

trong đó:

- $\mathcal{S}$ là tập trạng thái (state set), chứa tất cả trạng thái của môi trường cho trước;
- $\mathcal{A}$ là tập hành động (action set), chứa tất cả hành động mà agent có thể thực hiện;

- $\mathcal{T} : P(s_{t+1}|s_t, a_t)$ là transition function của môi trường, chỉ xác suất môi trường biến đổi từ trạng thái s_t sang trạng thái s_{t+1} khi agent thực hiện hành động a_t;
- $\mathcal{R} : P(r_t|s_t, a_t)$ là hàm reward, chỉ xác suất agent nhận reward r_t khi agent thực hiện hành động a_t trong trạng thái s_t.

Trong cài đặt này, giả sử trạng thái của môi trường được agent biết đầy đủ. Cài đặt này tạo thành một quá trình quyết định Markov đầy đủ (MDP) có thể quan sát được.

Hành vi của agent được mã hóa trongtrong chính sách hành vi: $\pi(a_t\ verts_t) = P(a_t|s_t)$ là phân bố xác suất trên các hành động sẵn có ở trạng thái nhất định của môi trường. Chất lượng của một chính sách được đo bằng lợi nhuận của nó, tức là tổng lợi ích mong đợi trong khoảng thời gian T:

$$G_t^\pi = \sum_{t=0}^{T} \gamma^t E[r_t],$$

trong đó $\gamma < 1$ là hệ số chiết khấu đánh giá thấp phần thưởng trong tương lai xa so với phần thưởng trong tương lai gần. Hệ số chiết khấu cung cấp một lời giải chính thức cho vấn đề tổng vô hạn (đối với $T \to \infty$) và trọng số trực quan khiến agent của chúng tôi ưu tiên các phần thưởng gần đúng thời điểm thay vì trì hoãn. Với khái niệm lợi nhuận này, mục đích của agent là tìm hiểu chính sách tối ưu nhằm tối đa hóa lợi nhuận G, đó là chính sách, không nhất thiết là duy nhất, sao cho không có chính sách nào khác tạo ra lợi nhuận cao hơn. Tìm hiểu một chính sách tối ưu yêu cầu agent phải cân bằng giữa động lực tìm hiểu (tìm kiếm các trạng thái và hành động chưa từng thấy trước đây mang lại phần thưởng cao) và việc khai thác (tham lam lựa chọn các trạng thái và hành động hiện được coi là để trả lại phần thưởng tốt nhất).

Tương tác với môi trường (và từ đó học tập) diễn ra qua các bước (steps) và giai đoạn (episodes). Một bước là sự tương tác của agent với môi trường: thực hiện một hành động duy nhất a_t theo chính sách π, thu thập phần thưởng r_t và quan sát môi trường phát triển từ trạng thái s_t sang trạng thái s_{t+1}. Một giai đoạn là tập hợp các bước từ trạng thái ban đầu đến trạng thái kết thúc.

Lưu ý rằng trong các tập khác nhau, ngay cả khi signature của RL có vấn đề $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$ không thay đổi, cách cài đặt có thể khác. Agent RL được đào tạo không chỉ để giải quyết một trường hợp cụ thể của một vấn đề mà là toàn bộ tập hợp các vấn đề có cấu trúc tương tự được mô tả bởi dạng $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$. Sự thay đổi giữa các giai đoạn này rất quan trọng và nó cho phép agent RL tổng quát hóa vấn đề.

2.2.2 Các thuật toán RL. Một số thuật toán đã được đề xuất để giải quyết vấn đề RL. Một trong những thuật toán RL đơn giản nhất nhưng hoạt động tốt là họ các phương pháp giá trị hành động (action-value method). Các thuật toán này giải quyết vấn đề tìm kiếm một chính sách tối ưu thông qua hàm proxy nhằm ước tính giá trị của từng cặp trạng thái và hành động.

Về mặt hình thức, các phương thức này xác định hàm giá trị hành động (action-value function) cho chính sách π là:

$$q^\pi(s_t, a_t) = E[G_t|s_t, a_t],$$

là hàm giá trị hành động cho một cặp (trạng thái, hành động) là lợi nhuận kỳ vọng từ trạng thái s_t sau khi thực hiện hành động a_t theo chính sách π. Nói chung, các phương pháp giá trị hành động tuân theo cách tiếp cận để học một chính sách tối ưu được gọi là lặp lại chính sách tổng quát dựa trên hai bước:

1) đưa ra chính sách khởi đầu π_0 tương tác với môi trường để tìm hiểu giá trị gần đúng của hàm $q^{\pi_0}(s_t, a_t)$;
2) cải thiện chính sách π_0 bằng cách xác định chính sách mới π_1 trong đó, ở mỗi trạng thái s_t, agent thực hiện hành động a_t tại đó tối đa hóa hàm giá trị hành động $q^{\pi_0}(st, a)$, nghĩa là

$$\pi^1(a_t|s) = \begin{cases} 1 & \text{nếu } a_t = \arg\max_a q^{\pi_0}(s_t, a) \\ 0 & \text{nếu ngược lại} \end{cases}$$

Lặp lại quá trình này (và cho phép không gian để tìm lời giải), agent cuối cùng sẽ hội tụ đến chính sách tối ưu π^*. Mặc dù bước thứ hai khá tầm thường, chỉ bao gồm một hoạt động tối đa hóa, nhưng bước đầu tiên yêu cầu phải điều chỉnh hàm giá trị hành động và có thể khó khăn hơn cũng như tốn thời gian hơn. Có hai cách chính để biểu diễn hàm giá trị hành động:

- Biểu diễn dạng bảng: cách biểu diễn này dựa trên ma trận hoặc tensor Q để mã hóa chính xác từng cặp (trạng thái, hành động) và ước tính giá trị của nó; cách biểu diễn dạng bảng đơn giản, dễ kiểm tra và hợp lý về mặt thống kê; tuy nhiên chúng có khả năng tổng quát hóa hạn chế và không có khả năng mở rộng tốt theo chiều của không gian trạng thái $\mathcal{S}$ và không gian hành động $\mathcal{A}$.
- Biểu diễn xấp xỉ: cách biểu diễn này dựa vào việc khớp một hàm gần đúng $\hat{q}$; các lựa chọn thông thường cho $\hat{q}$ là các hàm tham số từ hồi quy tuyến tính đơn giản đến các mạng nơ-ron phức tạp; các hàm gần đúng giải quyết vấn đề xử lý không gian trạng thái lớn $\mathcal{S}$ và không gian hành động $\mathcal{A}$, đồng thời cung cấp khả năng tổng quát hóa; tuy nhiên chúng khó diễn giải hơn và thường thiếu sự đảm bảo về mặt thống kê cho sự hội tụ.

2.2.3 Q-Learning. Thuật toán giá trị hành động tiêu chuẩn để giải quyết vấn đề RL là Q-learning. Q-learning là một thuật toán RL ngoại tuyến khác biệt theo thời gian (temporal-difference offpolicy); chênh lệch thời gian có nghĩa là thuật toán ước tính hàm giá trị hành động $q(s_t, a_t)$ bắt đầu từ dự đoán ban đầu (bootstrap) và cập nhật từng bước ước tính của nó có tham chiếu đến giá trị của các trạng thái và hành động trong tương lai; phi chính sách (offpolicy) có nghĩa là Q-learning có thể học một chính sách tối ưu trong khi tìm hiểu môi trường theo chính sách khác tối ưu π^*. Q-learning tạo nên một thuật toán linh hoạt cho phép giải quyết nhiều vấn đề RL; nó có thể được triển khai bằng cả cách biểu diễn dạng bảng Q của hàm giá trị hành động hoặc với cách biểu diễn gần đúng $\hat{q}(s_t, a_t)$.

Về mặt hình thức, đưa ra một bài toán RL $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$ với mức chiết khấu γ, một agent tương tác với môi trường

trong thời gian thực có thể dần dần xây dựng một giá trị gần đúng của hàm giá trị hành động thực $q(s_t, a_t)$ thông qua biểu diễn dạng bảng bằng cách cập nhật dần dần ước tính theo công thức:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_t + \gamma \max_x Q(s_{t+1}, x) - Q(s_t, a_t) \right]$$

trong đó $\alpha \in \mathbb{R}$ là đại lượng vô hướng xác định step-size. Một cách trực giác, ở mỗi bước, ước tính của $Q(s_t, a_t)$ sẽ di chuyển về phía hàm giá trị hành động thực tế $q(s_t, a_t)$ từng bước α theo cách giống ascent gradient.

3 MÔ HÌNH HÓA PT NHƯ MỘT BÀI TOÁN RL

3.1 PT được xem xét như Bài toán RL

Kiểm thử xâm nhập (Penetration Testing - PT), đặc biệt khi được khuôn mẫu như một vấn đề Capture The Flag (CTF), có thể được mô hình hóa như một trò chơi, làm cho nó phù hợp với các phương pháp Học Tăng Cường (Reinforcement Learning - RL). Thiết lập CTF cung cấp một khung làm việc rõ ràng với các người chơi có thể xác định được (đội đỏ và đội xanh), quy tắc (logic của hệ thống), và điều kiện chiến thắng (chiếm được cờ).

Trong bối cảnh này, vai trò của kẻ tấn công (đội đỏ) phù hợp với các giai đoạn thăm dò và khai thác phổ biến trong RL. Hành động thăm dò bao gồm việc thu thập thông tin, trong khi hành động khai thác bao gồm việc sử dụng thông tin đó để xâm nhập hệ thống. Do RL đã thành công trong việc giải quyết các trò chơi, việc xem xét nó cho PT là điều tự nhiên. Tuy nhiên, PT đặt ra các thách thức độc đáo khác với các trò chơi truyền thống, chẳng hạn như không gian hành động và trạng thái rộng lớn, và các kênh hạn chế để thu thập thông tin. Những thách thức này làm cho PT trở thành một vấn đề RL phức tạp hơn đơn thuần chỉ là tương tự trò chơi.

3.2 Cấu trúc của bài toán PT

Các RL agent giải quyết vấn đề bằng cách nhận diện và khai thác cấu trúc cơ bản. Các trò chơi thường có cấu trúc được định nghĩa rõ ràng, nhưng các hệ thống PT có thể che đậy logic của chúng, trình bày các mức độ cấu trúc khác nhau:

Hệ Thống Hoàn Hảo: Không có lỗ hổng, các biện pháp khai thác đều thất bại nên không có gì để học.

Hệ Thống Max-Entropy: Có lỗ hổng nhưng môi trường luôn thay đổi khiến kiến thức agent mới học được không thể sử dụng. Mỗi hành động của agent chỉ tiết lộ sự thành công hay thất bại của nó, cung cấp thông tin tối thiểu cho agent học hỏi. Tình huống này không thuận lợi cho việc học hiệu quả và bị loại khỏi xem xét.

Hệ Thống CTF: Có lỗ hổng và đủ cấu trúc để cho phép agent suy luận thông tin hữu ích từ các hành động của nó, làm cho chúng phù hợp với RL.

Hơn nữa, các hệ thống PT có thể không tĩnh, nghĩa là chúng có thể thay đổi để phản ứng với các hành động của kẻ tấn công, làm phức tạp quá trình học tập. Các môi trường không tĩnh yêu cầu agent phải thăm dò và thích nghi hiệu quả với các cấu trúc động. Các thuật toán RL hiệu quả cho PT cần nhấn mạnh việc thăm dò để khám phá các cấu trúc này.

3.3 Kiến thức tiên quyết trong cấu trúc bài toán PT

Để quản lý độ phức tạp của việc học trong PT, việc tiêm vào kiến thức tiên quyết có thể có lợi. Kiến thức này giúp đơn giản hóa quá trình học tập, làm cho nó hiệu quả hơn. Các kỹ thuật chính bao gồm:

Lazy Loading: Phương pháp thực nghiệm này khởi tạo các cặp (trạng thái, hành động) mới chỉ khi chúng được gặp, giả định rằng cấu trúc của vấn đề là thưa thớt. Điều này giảm bớt độ phức tạp của học dựa trên Q-table bằng cách tránh các đánh giá không cần thiết.

State Aggregation: Gộp các cặp (trạng thái, hành động) tương đồng về logic, tận dụng các lớp tương đương trên các trạng thái. Điều này nâng cao khả năng tổng quát hóa trong cả học dựa trên Q-table và học Q-nơ-ron, cho phép agent xử lý các trạng thái tương tự như nhau.

Imitation Learning: Định hình chính sách của agent bằng các mẫu hành vi hiệu quả cao từ con người hoặc các agent khác. Cách tiếp cận dựa trên dữ liệu này cung cấp các ví dụ ngầm truyền đạt cấu trúc của bài toán, giảm thời gian học cho cả học dựa trên Q-table và học Q-nơ-ron.

Các kỹ thuật này mặc dù có một mức độ chuyên môn vào thuật toán nhưng không yêu cầu một mô hình đầy đủ và rõ ràng của hệ thống mục tiêu. Chúng cung cấp một sự cân bằng giữa các phương pháp không mô hình hoàn toàn và có mô hình, nâng cao hiệu quả của các RL agent trong các kịch bản PT.

4 MÔ HÌNH HÓA BÀI TOÁN PT

4.1 Phân loại bài toán CTF

CTF có thể được phân loại theo loại lỗ hổng mà chúng thể hiện và loại khai thác mà người chơi dự kiến phải thực hiện. Mỗi loại vấn đề CTF có thể biểu hiện các dạng cấu trúc đặc trưng và có thể được mô hình hóa độc lập. Trong bài viết này, chúng ta sẽ xem xét các loại vấn đề CTF điển hình sau:

- **Port scanning and intrusion:** Trong vấn đề CTF này, một hệ thống máy chủ mục tiêu lộ ra trên mạng một tập hợp các cổng, và kẻ tấn công phải kiểm tra chúng, xác định một cổng có lỗ hổng và chiếm được cờ sau cổng đó bằng cách sử dụng một lỗ hổng đã biết.
- **Server hacking:** Trong vấn đề CTF này, một hệ thống máy chủ mục tiêu lộ ra trên mạng một tập hợp các dịch vụ, và kẻ tấn công phải tương tác với chúng, phát hiện một lỗ hổng, có thể là lỗ hổng không có tham số hoặc lỗ hổng có tham số, và chiếm được cờ bằng cách khai thác lỗ hổng đã phát hiện.
- **Website hacking:** Một loại con của server hacking, trong vấn đề CTF này, một hệ thống máy chủ mục tiêu lộ ra trên mạng một trang web, và kẻ tấn công phải kiểm tra các trang có sẵn, đánh giá xem có trang nào chứa lỗ hổng không, và chiếm được cờ sau một trong các trang đó bằng cách khai thác lỗ hổng đã phát hiện.

Ba loại này cung cấp các nhiệm vụ được biết đến rộng rãi và có thể được mô hình hóa như các vấn đề RL ở các mức độ đơn giản hóa và trừu tượng khác nhau.

4.2 Mô hình RL

Chúng ta xem một RL agent là một người trong red team tương tác với hệ thống mục tiêu có lỗ hổng. Hệ thống mục tiêu tạo thành môi trường mà agent tương tác. Mục tiêu của agent là chiếm được cờ trong môi trường mục tiêu một cách nhanh nhất có thể. Với vấn đề RL $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$, chúng ta đặt các yêu cầu và điều kiện sau:

Không gian trạng thái $\mathcal{S}$ được giả định là một tập hợp hữu hạn không cấu trúc các trạng thái mã hóa trạng thái của môi trường và, được ngầm hiểu là trạng thái kiến thức của agent.

Không gian hành động $\mathcal{A}$ được giả định là một tập hợp hữu hạn không cấu trúc chứa tất cả các hành động có thể được thực hiện bởi agent. Lưu ý rằng tập hợp này giống nhau ở mọi trạng thái; ngay cả khi một số hành động có thể không khả dụng cho agent ở một số trạng thái, thông tin này không được cung cấp cho agent; agent được mong đợi học từ kinh nghiệm để biết hành động nào khả dụng ở trạng thái nào.

Hàm chuyển tiếp $\mathcal{T}$ được giả định là một hàm mã hóa logic của kịch bản CTF cụ thể sẽ được xem xét.

Hàm thưởng $\mathcal{R}$ được giả định là một hàm thể hiện hiệu suất của agent. Agent nhận một phần thưởng âm nhỏ cho mỗi lần thử (thường là -1), và một phần thưởng lớn cho việc đạt được mục tiêu (thường là 100); cấu hình này sẽ thúc đẩy agent học chiến lược hiệu quả nhất (về mặt số lần thử) để chiếm cờ.

Trong phân tích thực nghiệm tiếp theo, chúng ta sẽ tập trung vào một thuật toán cụ thể, đó là học dựa trên Q-table (tabular Q-learning). Lý do cho lựa chọn này là: (i) học dựa trên Q-table là một thuật toán kinh điển và hiệu quả, cho phép chúng ta liên hệ kết quả của mình với tài liệu đã có; (ii) nó đảm bảo rằng agent sẽ hội tụ đến một chính sách tối ưu; (iii) việc sử dụng biểu diễn bảng cho phép diễn giải kết quả đơn giản hơn; (iv) học dựa trên Q-table có tốc độ và hiệu quả theo từng bước, do đó cho phép chúng ta dễ dàng lặp lại các thí nghiệm và đảm bảo tính tái lập; (v) học dựa trên Q-table có ít siêu tham số, cho phép điều chỉnh hiệu quả hơn. Nhược điểm chính của việc áp dụng học dựa trên Q-table là khả năng mở rộng, điều này ngầm giảm độ phức tạp của các vấn đề mà chúng ta có thể xem xét. Mặc dù có hạn chế này, kết quả vẫn sẽ chứng minh và xác nhận khả năng giải quyết các vấn đề CTF bằng RL, và cho phép chúng ta đánh giá tầm quan trọng của các thách thức mà chúng ta đã xác định.

5 PHÂN TÍCH THỰC NGHIỆM

Trong phần này, chúng tôi thực hiện lại 5 ví dụ của nhóm tác giả ban đầu, thực hiện phân tích thêm kết quả đạt được của agent khi thay đổi một số tham số, cũng như tạo thêm một ví dụ mô phỏng để đánh giá toàn diện hơn. Các ví dụ cụ thể về các thử thách CTF đơn giản được mô hình hóa dưới dạng các bài toán RL sử dụng các nguyên tắc đã thảo luận trong Mục 4, và được giải quyết bằng Q-learning. Chúng tôi xem xét các thử thách CTF với độ phức tạp tăng dần, và khi đối mặt với các thử thách đã xác định trong Mục 3, chúng tôi đánh giá các phương pháp khác nhau để giới thiệu kiến thức tiên nghiệm mà chúng tôi đã nhắc đến trong Mục 3. Tất cả các mô phỏng đều được triển khai theo giao diện RL chuẩn được định nghĩa trong thư viện OpenAI gym.

5.1 Simulation 1: Port Scanning CTF Problem

Trong mô phỏng này, chúng tôi xem xét một vấn đề *quét cổng* rất đơn giản được xây dựng trên loại bài toán CTF đầu tiên mà chúng tôi đã đề cập. Chúng tôi sử dụng thuật toán Q-learning cơ bản để giải quyết và phân tích kết quả khi so sánh với giải pháp tốt nhất và số bước để hội tụ. Mô phỏng cơ bản này nhằm mục đích cho thấy khả năng của các agent RL trong một kịch bản có cấu trúc môi trường hạn chế và ổn định.

Kịch bản CTF. Hệ thống mục tiêu là một máy chủ chỉ chạy một dịch vụ bị ảnh hưởng bởi một lỗ hổng đã biết. Số cổng mà dịch vụ chạy là không xác định; tuy nhiên, một khi cổng dịch vụ được phát hiện, agent biết chắc chắn nơi dịch vụ dễ bị tấn công và cách khai thác nó. Agent đội đỏ có thể tương tác với máy chủ bằng cách chạy quét cổng hoặc thực hiện khai thác tới một cổng cụ thể. Trong kịch bản đơn giản hóa này, lỗ hổng có thể được khai thác mà không cần tham số; cũng giả định rằng không có hành động nào được thực hiện bởi đội xanh trên hệ thống mục tiêu.

Thiết lập RL. Một máy chủ mục tiêu mở N cổng, mỗi cổng cung cấp một dịch vụ khác nhau; một trong số các dịch vụ bị ảnh hưởng bởi một lỗ hổng, và đằng sau nó là cờ mục tiêu (flag).

Chúng tôi mô hình hóa tập hành động $\mathcal{A}$ như một tập hợp gồm $N+1$ hành động: một hành động quét cổng và một hành động khai thác cho mỗi cổng trong số N cổng hiện có. Chúng tôi cũng mô hình hóa tập trạng thái $\mathcal{S}$ như một tập hợp các biến nhị phân $N+1$: một trạng thái ban đầu đại diện cho trạng thái hoàn toàn không biết của agent và một trạng thái cho mỗi cổng có giá trị là 1 khi agent phát hiện nó là cổng dễ bị tấn công. Kích thước của ma trận Q tăng theo $O(N^2)$.

Bài toán đơn giản này cho phép đánh giá cơ bản về khả năng học hỏi của agent. Lưu ý agent không được học giải pháp cho một trường hợp duy nhất của trò chơi CTF này mà phải học một chiến lược chung cho phép nó giải quyết vấn đề.

Về cơ bản, chiến lược tối ưu để giải quyết bài toán là một chính sách hai bước quét và sau đó khai thác cổng có lỗ hổng. Tuy nhiên, RL agent không nhận thức được ý nghĩa của các hành động có sẵn và nó không thể suy luận ra một chiến lược tối ưu, mà chỉ có thể học thông qua thử và sai.

Kết quả. Chúng tôi chạy mô phỏng của mình với thiết lập $N = 64$ cổng. Chúng tôi khởi tạo ngẫu nhiên chính sách của agent và chạy 1000 tập. Chúng tôi lặp lại mỗi mô phỏng 100 lần để thu thập số liệu thống kê đáng tin cậy.

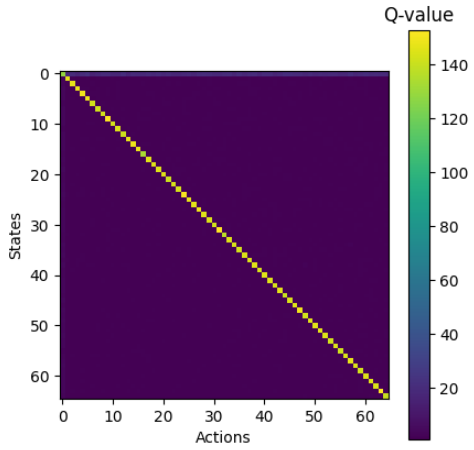

Figure 1: Ma trận giá trị hành động Q đã học được.

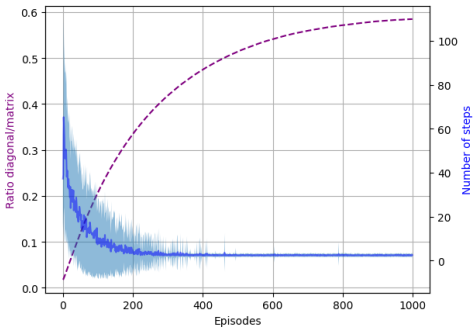

Figure 2: Tỷ lệ $\frac{\sum_{i=0}^{N} Q_{ii}}{\sum_{i,j=0}^{N} Q_{ij}}$ theo số tập (màu tím) và số bước cần để lấy flag theo số tập (màu xanh).

Hình 1 cho thấy ma trận giá trị hành động Q đã học được. Hình 2 vẽ tỷ lệ $\frac{\sum_{i=0}^{N} Q_{ii}}{\sum_{i,j=0}^{N} Q_{ij}}$ theo số tập (màu tím), và số bước agent cần để lấy flag theo số tập (màu xanh). Vì chúng tôi biết rằng chiến lược tối ưu được mã hóa dọc theo đường chéo chính của Q, thống kê này cho chúng tôi biết bao nhiêu trọng số của Q được phân bố dọc theo đường chéo. Sau khoảng 400 tập, việc học của agent bước vào giai đoạn bão hòa. Lưu ý rằng tỷ lệ này chỉ hội tụ về 1 khi tiến đến dương vô cùng. Đường màu xanh trong Hình 2 minh họa số bước mỗi tập. Sau khoảng 200 tập, agent đã học chiến lược tối ưu và hoàn thành các thử thách trong số hành động tối thiểu.

Nhận xét. Thành công của RL trong mô phỏng này không có gì ngạc nhiên; tuy nhiên, nó nêu lên những thách thức cụ thể khi sử dụng RL để giải quyết các vấn đề hacking: giải quyết thách thức CTF đòi hỏi học cấu trúc của vấn đề; điều này có thể thực hiện được, nhưng sử dụng dữ liệu trải nghiệm và suy luận có nghĩa là agent RL phải phụ thuộc mạnh mẽ vào sự khám phá. Gần hai trăm tập là cần thiết để hội tụ đến một giải pháp, một số lần thử nhiều hơn đáng kể so với những gì cần thiết cho một người trong đội đỏ để tìm ra chiến lược tối ưu.

5.2 Simulation 2: Non-stationary Port-scanning CTF Problem

Kịch bản CTF. Trong kịch bản này, đội xanh không còn bị động mà có thể phản ứng lại các hành động của đội đỏ. Chúng tôi thiết lập hệ thống mục tiêu như trước: máy chủ có một dịch vụ dễ bị tấn công chạy trên một cổng mà số cổng này không được biết trước với kẻ tấn công. Để mô phỏng kịch bản tấn công-phòng thủ, chúng tôi giả định rằng đội xanh nhận thức được dịch vụ dễ bị tấn công nhưng không thể ngăn chặn nó vì điều này sẽ ảnh hưởng đến hoạt động kinh doanh liên tục của họ. Đội xanh không thể lọc lưu lượng, và lựa chọn duy nhất họ có là di chuyển dịch vụ sang cổng khác nếu họ quan sát thấy các hành động có thể báo hiệu một cuộc tấn công. Trường hợp này khá phi thực tế, nhưng chúng tôi sử dụng nó như một cuộc thi tấn công-phòng thủ đơn giản với các hành động giới hạn.

Thiết lập RL. Chúng tôi xem xét kịch bản quét cổng như đã định nghĩa trong mô phỏng trước, và thêm vào một yếu tố: bất cứ khi nào kẻ tấn công sử dụng hành động quét cổng, máy chủ mục tiêu sẽ phát hiện với xác suất p; nếu phát hiện thành công, cờ sẽ được di chuyển ngẫu nhiên sang một cổng mới. Với môi trường không cố định này, vấn đề này trở thành một bài toán học tập khó khăn hơn so với trước. Đặc biệt, kiến thức về cấu trúc mà agent có được thông qua việc quét cổng có thể không đáng tin cậy. Trong bối cảnh ngẫu nhiên này, chiến lược tối ưu không nhất thiết phải là chính sách quyết định như trong Mô phỏng 1.

Kết quả. Chúng tôi chạy mô phỏng của mình với $N = 16$ cổng. Tất cả các tham số còn lại của mô phỏng này đều giống như trong Mô phỏng 1. Chúng tôi xem xét tất cả các giá trị có thể của p trong tập $\{0, 0.1, 0.2, ..., 0.9, 1.0\}$. Chúng tôi lặp lại mỗi mô phỏng 100 lần để thu thập thống kê đáng tin cậy.

Hình 3 trình bày các ma trận giá trị hành động Q học được với $p = 0.1$, $p = 0.5$ và $p = 1$. Với giá trị p nhỏ, ma trận Q giống với mẫu quan sát được trong Mô phỏng 1, đối với các giá trị cao hơn của p ma trận Q dần mất cấu trúc này. Trong trường hợp $p = 0.1$ (Hình 3(a)), việc sử dụng hành động quét cổng ở đầu, tiếp theo là hành động khai thác có xác suất thành công cao là hợp lý; do đó ta thấy dạng đường chéo thông thường. Trong trường hợp ngẫu nhiên hơn $p = 0.5$ (Hình 3(b)), có 50% khả năng hành động quét cổng bị phát hiện và cờ được di chuyển; tuy nhiên, sử dụng hành động quét cổng và khai thác mục tiêu vẫn là một nước cược hợp lý, mặc dù kém hiệu quả hơn. Cuối cùng, trong trường hợp hoàn toàn ngẫu nhiên $p = 1$ (Hình 3(c)), hành động quét cổng chắc chắn bị phát hiện, và không có kế hoạch nào có thể được xây dựng dựa trên thông tin thu thập được; agent cơ bản phải dựa vào việc đoán ngẫu nhiên. Hình 4 cho thấy số bước agent cần để lấy flag theo mỗi tập khi $p = 0.1$, $p = 0.5$, $p = 1$.

Với sự tăng dần của p, số bước agent cần tăng lên vì nó phải cố gắng đoán vị trí của lỗ hổng. Để ý ta thấy rằng số bước trung bình trong trường hợp $p = 1$ nhiều hơn số cổng; điều này là do agent thỉnh thoảng thử hành động quét cổng, dẫn đến việc cờ bị di chuyển, và yêu cầu agent thử lại khai thác trên các cổng đã kiểm tra.

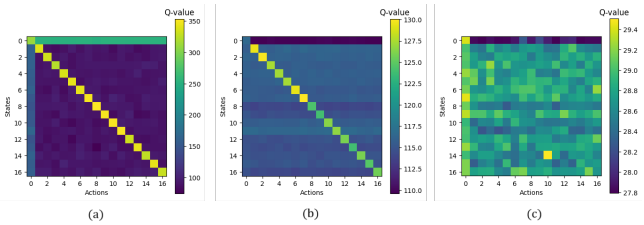

Figure 3: Kết quả của Mô phỏng 2. Ma trận giá trị hành động Q học được với: (a) $p = 0.1$, (b) $p = 0.5$, và (c) $p = 1$.

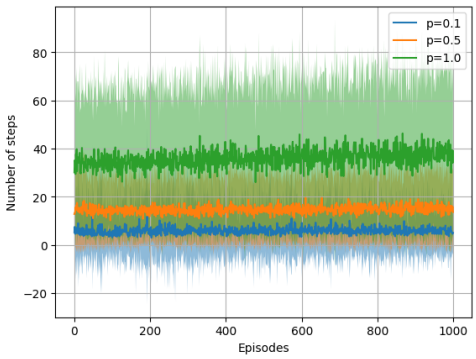

Figure 4: Kết quả của Mô phỏng 2. Số bước cần để lấy flag theo số tập với $p = 0.1$, $p = 0.5$, $p = 1$.

Nhận xét. Việc khiến môi trường có cấu trúc thay đổi làm cho vấn đề trở nên khó khăn hơn, bằng cách ngăn cản agent học chính xác cấu trúc của vấn đề với sự chắc chắn. Mặc dù vậy, một agent học Q-learning vẫn có thể giải quyết vấn đề CTF này theo cách hợp lý, nhưng không tối ưu, tùy thuộc vào mức độ ngẫu nhiên và không cố định. Đối với $p = 0$, hệ thống CTF có cấu trúc rõ ràng và agent có thể học được chính sách tối ưu; khi p tăng, môi trường dần dần chuyển từ hệ thống CTF sang hệ thống max-entropy (xem Mục 3). Khi đó không có hành động nào cung cấp thông tin thực tế về cấu trúc của máy chủ mục tiêu (hành động quét cổng thu được thông tin không đáng tin cậy và vô ích); không thể tìm ra bất kỳ cấu trúc nào, RL có rất ít tác dụng: tất cả những gì có thể làm là đoán, tức là thử từng cổng một để tìm lỗ hổng. Điều này nêu bật vai trò của cấu trúc trong việc học sử dụng các agent RL.

5.3 Simulation 3: Server Hacking CTF Problem with Lazy Loading

Trong mô phỏng này, chúng tôi xem xét một vấn đề phức tạp gần thực tế hơn đại diện cho một kịch bản tấn công máy chủ đơn giản. Mặc dù được đơn giản hóa rất nhiều, vấn đề này đã tạo ra một thách thức nghiêm trọng cho agent Q-learning dạng bảng do kích thước lớn của bảng Q. Để giải quyết trở ngại này, chúng tôi dựa vào kiến thức tiên nghiệm dưới dạng lazy loading để kiểm soát được kích thước của không gian trạng thái và hành động và loại bỏ các trạng thái không liên quan. Chúng tôi phân tích cách agent học trong kịch bản này và ảnh hưởng của phương pháp đã áp dụng.

Kịch bản CTF. Trong mô phỏng này, một server mục tiêu cung cấp các dịch vụ tiêu chuẩn khác nhau, như web, FTP hoặc SSH. Mỗi dịch vụ có thể có lỗ hổng, hoặc là lỗ hổng đơn giản dễ khai thác mà không cần tham số (như một trang Wordpress với một plugin có thể dẫn đến tiết lộ thông tin tại một URL cụ thể) hoặc là lỗ hổng yêu cầu kẻ tấn công phải gửi đầu vào đặc biệt (như một plugin Wordpress với SQL injection).

Kẻ tấn công có thể thực hiện ba loại hành động thu thập thông tin:

(1) Kiểm tra các cổng mở và dịch vụ trên server.
(2) Thử tương tác với các dịch vụ bằng các giao thức đã biết để thu được thông tin cơ bản và phát hiện lỗ hổng đã biết.
(3) Tương tác chi tiết hơn với các thiết lập dịch vụ độc đáo hoặc trang web tùy chỉnh để xác định các lỗ hổng chưa được ghi nhận và các tham số đầu vào cần thiết cho khai thác.

Agent cũng có hai hành động khai thác:

(1) Khai thác lỗ hổng không tham số bằng cách truy cập dịch vụ có lỗ hổng và lấy cờ.
(2) Chọn một tham số từ một tập hợp định trước và gửi nó đến dịch vụ để khai thác lỗ hổng có tham số và lấy cờ.

Thiết lập RL. Chúng tôi định nghĩa một server mục tiêu mở N cổng, mỗi cổng cung cấp một trong V dịch vụ khác nhau. Một trong các dịch vụ có lỗ hổng, và đằng sau nó là cờ mục tiêu. Lỗ hổng có thể là lỗ hổng không tham số hoặc lỗ hổng có tham số. Trong trường hợp sau, lỗ hổng có thể đã được biết trước hoặc cần phải được phân tích sâu hơn. Tham số cho lỗ hổng có tham số được chọn từ một tập hợp M tham số khả thi.

Bộ hành động cơ bản có sẵn cho agent tạo ra một tập hợp A lớn hơn gồm các hành động cụ thể, mỗi loại hành động được thực hiện trên một cổng cụ thể. Tập hợp trạng thái S cũng có độ lớn lớn, do vấn đề theo dõi những gì agent đã học được trong quá trình tương tác với server. Trong bài báo gốc, các tác giả đã ước lượng số lượng trạng thái tổng thể bằng công thức như sau:

$$|S| \approx 2^{11} N^3 V M$$

Một agent Q-learning dạng bảng thuần túy sẽ cần một lượng bộ nhớ cực lớn chỉ để khởi tạo bảng Q của nó. Tuy nhiên, chỉ dựa vào kiến thức đơn giản rằng một số cặp (trạng thái, hành động) có thể không liên quan hoặc không mang tính thông tin, lazy loading được áp dụng: thay vì khởi tạo ngay từ đầu một ma trận giá trị hành động lớn không thể quản lý Q, ta dần dần xây dựng bảng Q dựa trên các cặp (trạng thái, hành động) mà agent gặp.

Kết quả. Chúng tôi chạy mô phỏng với N = 4 cổng, V = 5 dịch vụ, M = 4 tham số, khởi tạo ngẫu nhiên các cặp (trạng thái, hành động) trong thời gian chạy và chạy agent trong 10^6 tập. Mô phỏng được lặp lại 20 lần để thu thập số liệu thống kê tin cậy.

Hình 5 thể hiện số bước mà agent đã thực hiện để hoàn thành một nhiệm vụ. Hình 6 thể hiện phần thưởng mà agent nhận được. Chúng ta có thể thấy sự cải thiện của agent trong việc học. Phương sai cao được ghi lại một phần là do chỉ định

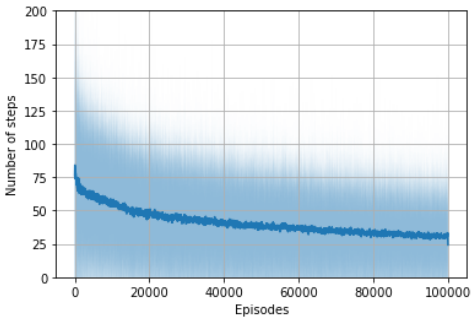

Figure 5: Hình minh họa số bước agent cần cho mỗi tập

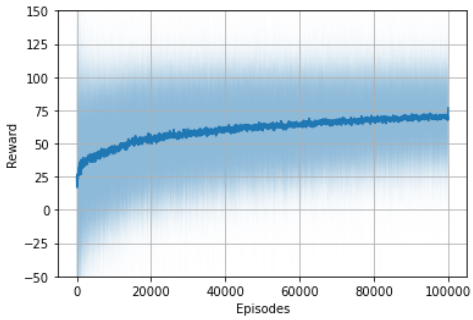

Figure 6: Hình minh họa phần thưởng agent nhận được mỗi tập

tính khám phá cao của agent (ϵ= 0.3) khiến agent thực hiện hành động ngẫu nhiên gần một phần ba số lần. Giới hạn trên của đường cong phần thưởng đạt đến mức từ 80 trở lên, chỉ ra rằng agent thực sự có thể học được một chiến lược hợp lý vì nó có thể giải quyết vấn đề CTF chỉ bằng một vài hành động so với số lượng lớn các hành động.

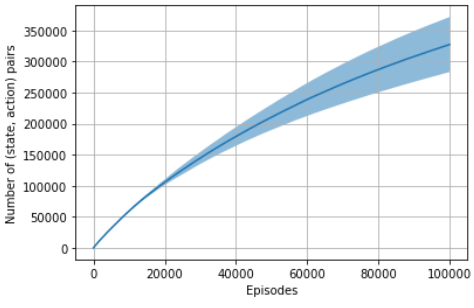

Figure 7: Số mục trong bảng Q của agent theo theo tập

Hình 7 cho thấy số lượng mục trong bảng giá trị hành động Q qua các tập. Đường cong có dạng parabol tăng nhanh ở phần đầu và chậm dần về cuối. Việc này là hợp lý vì lúc đầu mọi trạng thái mà agent gặp phải đều mới và cần được thêm vào bảng Q. Sử dụng công thức ước tính sơ bộ tổng số trạng thái có thể có $|\mathcal{S}|$ lớn hơn $2 * 10^6$, so với số lượng trạng thái agent đã học nhỏ hơn một bậc ($3.5 * 10^5$) và điều này chứng minh lazy loading cho phép agent học nhanh chóng một chính sách hợp lý với mức tiêu thụ bộ nhớ nhỏ hơn đáng kể.

Nhận xét. Mô phỏng này trình bày cách sử dụng lazy loading để giảm độ phức tạp của bài toán server hacking, giúp agent RL học được chính sách hiệu quả với chi phí tính toán hợp lý. Lazy loading đã chứng minh hiệu quả trong việc kiểm soát không gian trạng thái và hành động, cho phép agent tập trung vào các trạng thái và hành động liên quan nhất để đạt được mục tiêu.

5.4 Simulation 4: Website Hacking CTF Problem with State Aggregation

Lazy loading là một kỹ thuật đơn giản để đưa kiến thức tiên nghiệm có thể cho phép agent học hiệu quả trong môi trường phức tạp vừa phải bằng cách bỏ qua các cặp (trạng thái, hành động) không liên quan. Tuy nhiên, tính hữu ích của Lazy loading bị hạn chế, vì trong các môi trường phức tạp vốn có với số lượng lớn các trạng thái tương tự, số lượng cặp (trạng thái, hành động) được ghi lại có thể không thể quản lý được.

Trong mô phỏng trước Lazy loading đã được áp dụng nhưng số lượng (state, action) vẫn nhiều ($|\mathcal{S}| > 350000$ trong khi số lượng cổng, dịch vụ, tham số đều không quá 5).

Trong mô phỏng này, chúng tôi giới thiệu một phương pháp khác để quản lý độ phức tạp của không gian trạng thái và hành động, đó là việc sử dụng state aggregation. Mô phỏng này mô tả một kịch bản tấn công trang web, nơi agent phải học cách khai thác một lỗ hổng trên một trang web có nhiều trang và tham số khác nhau.

Kịch bản CTF. Trong mô phỏng này, chúng tôi giả định rằng kẻ tấn công biết vị trí của trang web mục tiêu, do đó không cần quét cổng hoặc nhận dạng giao thức. Trang web bao gồm một tập hợp các tệp: bắt đầu từ tệp chỉ mục index.html, kẻ tấn công có thể ánh xạ các tệp hiển thị bằng cách đọc nội dung HTML và bằng cách nhấp vào các liên kết bên trong nội dung. Trang web cũng có thể lưu trữ các tệp ẩn không được liên kết với chỉ mục. Một số tệp chứa các tập lệnh phía máy chủ và kẻ tấn công có thể xác định các đầu vào tùy chỉnh có thể được gửi để thực hiện việc khai thác và bắt cờ. Kẻ tấn công được cung cấp ba loại hành động thu thập thông tin: (i) Đọc tệp chỉ mục, tìm các liên kết và có được tất cả các tệp được liên kết trên máy chủ. (ii) Phân tích nội dung của một tập tin hiển thị và tìm ra tập tin ẩn. (iii) Phân tích một tập tin để tìm các tham số đầu vào có thể được sử dụng để khai thác. Và có thể thực hiện một hành động khai thác: (i) Gửi tham số đầu vào tới tệp có lỗ hổng và nếu chính xác thì sẽ lấy được cờ.

Thiết lập RL. Server host một trang web chứa N trang khác nhau, N_vis trang hiển thị có thể truy cập qua Internet, và N_his trang ẩn được liên kết tới các trang hiển thị. Một trong các trang chứa lỗ hổng bảo mật và có flag. Tham số cho lỗ hổng có tham số được chọn từ một tập hợp M tham số khả thi. Agent cần phải điều hướng qua các trang, xác định lỗ hổng và khai thác chúng để lấy được cờ. Các hành động của agent bao gồm truy cập các trang khác nhau, gửi các yêu cầu với các tham số đầu vào và phân tích phản hồi để xác định xem trang có lỗ hổng hay không.

Chúng tôi sử dụng kỹ thuật tổng hợp trạng thái (state aggregation). Thay vì xem xét từng trạng thái của mỗi trang web, chúng tôi nhóm các trạng thái tương tự lại với nhau.

Theo đó, thay vì yêu cầu agent học một chiến lược cụ thể trên mỗi tệp, ta hướng dẫn agent học một policy duy nhất sẽ được sử dụng trên tất cả các tệp. Tại mỗi thời điểm, agent sẽ chỉ tập trung vào một tệp duy nhất, tương tác với tệp đó và cập nhật policy toàn cục hợp lệ cho bất kỳ tệp nào. Điều này giúp giảm đáng kể kích thước của không gian trạng thái và giúp agent học hiệu quả hơn.

Khi áp dụng State aggregation, không gian action của agent đã giảm xuống đáng kể chỉ còn 6 hành động:

+ Scan: truy xuất đồ thị liên kết các tệp hiển thị
+ Explore: 1 file cụ thể để tìm file ẩn liên kết với file đó.
+ Inspect: 1 file cụ thể để tìm lổ hỗng tham số (nếu có)
+ Send: 1 file cụ thể để gửi payload khai thác lổ hổng
+ Focus Next: di chuyển sang file tiếp theo
+ Focus Prev: lùi về file trước

Kết quả. Chúng tôi thực hiện mô phỏng với $2 \leq N_{vis} \leq 4$ và $0 \leq N_{hid} \leq 2$, $M = 4$ tham số, khởi tạo ngẫu nhiên các cặp (trạng thái, hành động), áp dụng Lazy loading và State aggregation. Agent được chạy trong 10^5 tập với mức khám phá $\epsilon = 0,3$, sau đó thử nghiệm trên 100 tập với $\epsilon = 0$ để lấy kết quả đánh giá. Mô phỏng được thực hiện 20 lần để có được số liệu tin cậy.

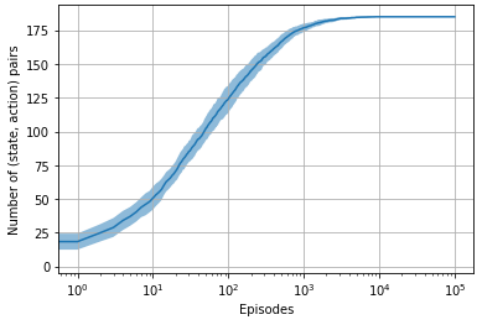

Figure 8: Kết quả của Mô phỏng 4. Số mục trong Q-table.

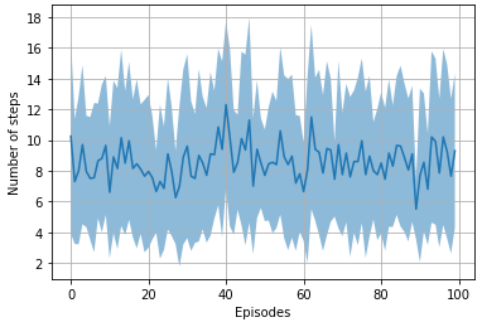

Figure 9: Kết quả của Mô phỏng 4. Số bước agent đã thực hiện để lấy flag.

Hình 8 hiển thị số cặp (trạng thái, hành động) trong bảng Q của các agent trong 10^5 tập. Số lượng trạng thái đạt đỉnh khi số tập chưa đạt đến 10^4, có khoảng 180 trạng thái mà agent gặp phải. Hình 9 thể hiện số bước trên 100 tập tiếp theo khi chạy các agent với tham số khám phá (ϵ) được đặt thành 0. Số bước dao động trong khoảng từ 8 đến 10, thực sự là số lượng cần thiết để thăm dò máy chủ mục tiêu, thu thập thông tin trên các tệp và cuối cùng lấy cờ.

Nhận xét. Mô phỏng này có mức độ phức tạp tương đương với Mô phỏng 3, tuy nhiên nhờ áp dụng State Agreegation không gian state đã giảm đáng kể từ 350,000 giảm xuống còn 180 state. Khi bài toán được đơn giản hóa, việc học của agent trở nên rất đơn giản và nhanh chóng. Điều đó cho thấy cách sử dụng các thuật toán và kỹ thuật RL thích hợp như Lazy loading và State agreegation có thể giúp agent RL giải quyết một cách hiệu quả một bài toán CTF đầy thách thức.

Kết quả từ mô phỏng này cho thấy việc sử dụng state aggregation giúp agent học nhanh hơn và hiệu quả hơn so với khi không sử dụng phương pháp này. State aggregation cho phép gom nhóm các trạng thái tương tự lại với nhau, giảm bớt số lượng trạng thái cần phải quản lý và tính toán. Điều này đặc biệt hữu ích trong các kịch bản có không gian trạng thái lớn và phức tạp như trong tấn công trang web.

Các tác giả trong bài báo gốc đã nhận xét: "*Việc tổng hợp trạng thái cho phép chúng tôi đưa ra một dạng kiến thức mà agent RL thường không có. Một người của đội đỏ có thể đi đến kết luận rằng việc hành động theo cách thống nhất với các tệp khác với kinh nghiệm trước đây của cô ấy với các tệp là hợp lý; kiến thức này cung cấp cho cô một con đường tắt hiệu quả để đạt được giải pháp. agent RL không có khả năng tương tự vì nó không có khái niệm chính thức về tệp; cuối cùng nó có thể học bằng cách suy luận một chính sách thực sự thống nhất cho tất cả các tệp, nhưng điều này đòi hỏi phải thu thập một lượng lớn kinh nghiệm.*"

Việc sử dụng cấu trúc đồ thị để thể hiện hệ thống tệp trên trang web mục tiêu cũng là một điểm thú vị, cách thao tác và khám phá biểu đồ thông minh hơn và phức tạp hơn có thể được xem xét để khai thác kiến thức về cấu trúc này và cải thiện hiệu suất của agent.

5.5 Simulation 5: Server Hacking CTF Problem with Imitation Learning

Lazy loading và State aggregation cải thiện hiệu suất của agent bằng cách giảm kích thước của không gian trạng thái; tuy nhiên, nếu agent phải khám phá một không gian trạng thái lớn và giải pháp tối ưu (hoặc thỏa đáng) chỉ chiếm một phần nhỏ trong không gian này thì quá trình tìm kiếm có thể mất rất nhiều thời gian. Trong mô phỏng này, chúng tôi xem xét cách định hướng quá trình học tập của agent bằng cách sử dụng phương pháp Imitation learning (học bắt chước).

Kịch bản CTF. Sử dụng lại bài toán Server hacking trong Mô phỏng 3. Trong kịch bản này, server mục tiêu có một số dịch vụ mở và một trong số chúng có lỗ hổng bảo mật. Agent phải phát hiện và khai thác lỗ hổng để lấy được cờ. Tuy nhiên, thay vì tự mình khám phá, agent được cung cấp một tập dữ liệu các hành động của chuyên gia trong quá trình tấn công server.

Quá trình học hỏi của agent bao gồm hai giai đoạn:

- Giai đoạn đầu, agent học từ các hành động của chuyên gia.

- Giai đoạn sau, agent tinh chỉnh chính sách của mình thông qua quá trình học reinforcement learning tiêu chuẩn.

Thiết lập RL. Sử dụng thiết lập được sử dụng trong Mô phỏng 3, nhưng áp dụng thêm kĩ thuật Imitation learning: agent được cung cấp một tập hợp D ví dụ mã hóa hành vi của một người chơi giả định trong đội đỏ đang cố gắng giải quyết bài toán CTF. Các ví dụ này thể hiện các mẫu hành vi thành công và cung cấp thông tin cho agent RL về mức độ liên quan của các tùy chọn khác nhau. Agent, thay vì bắt đầu trong trạng thái hoàn toàn không biết gì, lại được đưa ra các ví dụ về cách có thể kết hợp các hành động để đạt được giải pháp cho vấn đề. Điều này giúp đơn giản hóa vấn đề thăm dò: thay vì tìm kiếm trong toàn bộ không gian chính sách, agent sẽ tìm kiếm thiên về các chính sách do chuyên gia xác định. Sự thiên vị này cho phép giải quyết vấn đề hiệu quả hơn, nhưng nó cũng làm cho agent ít có khả năng phát hiện ra các chính sách khác biệt đáng kể với hành vi của con người.

Kết quả. Chúng tôi chạy mô phỏng bằng cách sử dụng cùng cài đặt được sử dụng trong Mô phỏng 3. Đầu tiên, chúng tôi đào tạo một agent RL tiêu chuẩn trong 10^5 tập. Sau đó, chúng tôi đào tạo ba agent học tập bắt chước, mỗi agent được cung cấp lần lượt 100, 200 và 500 bản trình diễn (chuỗi hành động lấy được flag); sau đó ba agent học bắt chước được đào tạo thêm trong 100 tập. Mô phỏng được lặp lại 20 lần để thu thập số liệu thống kê tin cậy.

Hình 10 cho thấy phần thưởng mà các agent khác nhau nhận được. Các đường gạch ngang biểu thị phần thưởng trung bình mà agent học bắt chước thu được trong 100 đợt đào tạo (những đường này độc lập với tỷ lệ trên trục x và được vẽ dưới dạng hằng số để so sánh). Đường màu xanh lam thể hiện giá trị điểm trung bình của 20 agent RL tiêu chuẩn nhận được trong quá trình đào tạo. Biểu đồ cho thấy agent RL tiêu chuẩn cần được đào tạo từ 3000 đến 6000 tập để đạt được phần thưởng tương đương với agent học tập bắt chước có thể đạt được với vài trăm lần học từ ví dụ; Ta cũng thấy rằng phần thưởng của các agent học bắt chước đạt được không quá chênh lệch nhau, khoảng cách điểm giữa agent học từ 100 ví dụ và agent học 500 ví dụ không đáng kể. Tuy nhiên khi xét đến mức độ ổn định, thì agent học từ 500 ví dụ ổn định hơn agent học từ 100 ví dụ, tương tự với agent học từ 200 ví dụ. Điều đó được thể hiện qua độ lệch chuẩn tính được cho trong bảng 1. Bằng cách áp dụng học bắt chước, mặc dù phần thưởng tổng thể vẫn chưa đạt đến mức tối ưu, nhưng nó cho phép thời gian đào tạo agent giảm đi một mức độ lớn.

Table 1: Kết quả của Mô phỏng 5. Độ lệch chuẩn của phần thưởng agent đạt được tương ứng với D ví dụ được học

D=100	ystd=10.611362824821324
D=200	ystd=8.817445931220673
D=500	ystd=5.234298520336798

Nhận xét. Không gian trạng thái và hành động trong bài toán này có kích thước lớn, nhưng học bắt chước (imitation learning) giúp agent bắt đầu với một chính sách gần đúng tốt, từ đó giảm thời gian học hỏi và cải thiện hiệu quả khai thác. Kết quả của mô phỏng này cho thấy rằng imitation learning giúp agent đạt được hiệu suất cao hơn nhanh chóng so với việc chỉ sử dụng reinforcement learning. Agent có thể học cách khai thác lỗ hổng từ dữ liệu chuyên gia và sau đó cải thiện chính sách của mình thông qua quá trình học reinforcement learning. Điều này chứng minh rằng imitation learning là một phương pháp hữu hiệu để giải quyết các bài toán CTF phức tạp.

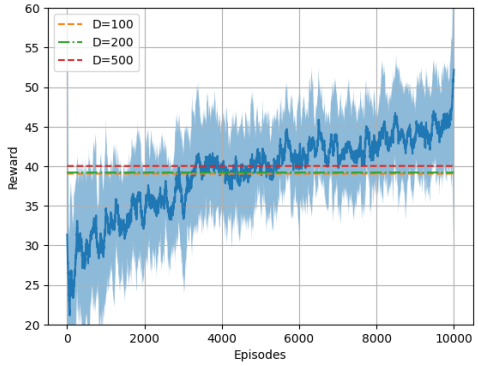

Figure 10: Kết quả của Mô phỏng 5. So sánh hiệu suất của agent khi sử dụng imitation learning và khi không sử dụng.

5.6 Simulation 6: Website Hacking CTF Problem with Imitation Learning

Đến đây, chúng tôi đã thực hiện 5 mô phỏng đi qua tất cả các dạng bài toán CTF cơ bản cũng như đã áp dụng các kĩ thuật cung cấp kiến thức tiên quyết cho agent và đánh giá hiệu quả của chúng. Tuy nhiên vẫn chưa có mô phỏng nào thực hiện đầy đủ cả 3 phương pháp Lazy loading, State agreegation và Imitation learning. Vì vậy trong mô phỏng này, chúng tôi áp dụng cả 3 phương pháp trên để đánh giá khả năng kết hợp của các phương pháp và mức độ mà chúng thúc đẩy agent học được chính sách tối ưu một cách nhanh chóng.

Kịch bản CTF. Sử dụng lại bài toán Website hacking trong Mô phỏng 4. Server mục tiêu host một trang web có một số trang public và một số trang ẩn và một trong số chúng có lỗ hổng. Agent phải phát hiện và khai thác lỗ hổng để lấy được cờ. Tuy nhiên, thay vì tự mình khám phá, agent được cung cấp một tập dữ liệu các hành động của chuyên gia trong quá trình tấn công server.

Quá trình học hỏi của agent bao gồm hai giai đoạn:

- Giai đoạn đầu, agent học từ các hành động của chuyên gia.
- Giai đoạn sau, agent tinh chỉnh chính sách của mình thông qua quá trình học reinforcement learning tiêu chuẩn.

Thiết lập RL. Sử dụng thiết lập được sử dụng trong Mô phỏng 4, đã áp dụng Lazying loading và State aggregation và áp dụng thêm kĩ thuật Imitation learning. Agent được cung cấp một tập hợp D ví dụ mã hóa hành vi của một người chơi giả định trong đội đỏ đang cố gắng giải quyết bài

toán CTF hack web. Các ví dụ này thể hiện các mẫu hành vi thành công và cung cấp thông tin cho agent RL về mức độ liên quan của các tùy chọn khác nhau. Áp dụng Imitation learning giúp giải quyết vấn đề hiệu quả hơn nhưng nó cũng hạn chế khả năng phát hiện ra hướng đi mới lạ của agent.

Kết quả. Chúng tôi chạy mô phỏng bằng cách sử dụng cùng cài đặt được sử dụng trong Mô phỏng 4, và áp dụng cùng chiến thuật Imitation learning trong mô phỏng 5. Đầu tiên, chúng tôi đào tạo một agent RL tiêu chuẩn trong 10^5 tập. Sau đó, chúng tôi đào tạo ba agent học tập bắt chước, mỗi agent được cung cấp lần lượt 100, 200 và 500 chuỗi hành động ví dụ; sau đó ba agent học bắt chước được đào tạo thêm trong 100 tập. Mô phỏng được lặp lại 20 lần để thu thập số liệu thống kê tin cậy.

Hình 11 cho thấy phần thưởng mà các agent khác nhau nhận được. Các đường gạch ngang biểu thị phần thưởng trung bình mà agent học bắt chước thu được trong 100 đợt đào tạo (những đường này độc lập với tỷ lệ trên trục x và được vẽ dưới dạng hằng số để so sánh). Đường màu xanh lam thể hiện giá trị điểm trung bình của 20 agent RL tiêu chuẩn nhận được trong quá trình đào tạo.

Chúng ta có thể đưa ra nhận xét rằng Imitation learning mặc dù có hiệu quả nhưng ảnh hưởng của nó không lớn, các agent được học trước mẫu mau chóng bị standard RL agent vượt qua sau chỉ sau vài trăm tập. Điều này có thể do không gian state, action nhỏ với chỉ 6 action và khoảng 180 state, dẫn đến việc agent tìm được chính sách tối ưu nhanh chóng. Một lý do khác cũng ảnh hưởng mạnh đến kết quả là do mức khám phá của agent đang được thiết lập khá cao ($\epsilon = 0.3$), do đó gần một phần ba số hành động của agent được chọn bất kỳ mà không tuân theo chính sách đã được học dẫn đến kết quả thấp hơn. Luận điểm này được xác thực qua hình 12. Có thể thấy với mức khám phá ϵ càng cao thì điểm thưởng của agent càng giảm. Tốc độ học α cũng là một yếu tố có ảnh hưởng lớn, hình 13 thể hiện ảnh hưởng của tốc độ học α tới phần thưởng của agent. Giá trị α càng cao, điểm thưởng của agent nhận được càng giảm.

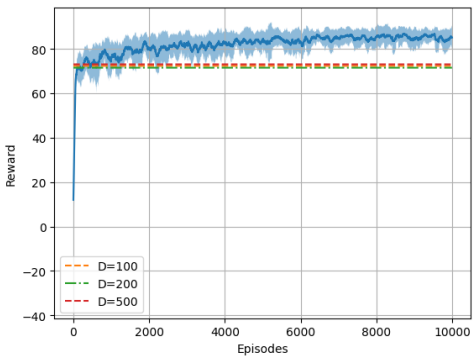

Figure 11: Kết quả của Mô phỏng 6. So sánh hiệu suất của agent khi sử dụng imitation learning và khi không sử dụng.

Nhận xét. Các biện pháp cung cấp kiến thức tiên nghiệm cho agent được xem xét có thể kết hợp với nhau để thúc đẩy agent học được chính sách tối ưu nhanh hơn mặc dù có hiệu quả nhưng tính hiệu quả của các phương pháp bị giảm và không gây ấn tượng mạnh như khi áp dụng một mình. Phương pháp Imitation learning có hiệu quả tốt với bài toán có không gian state, action lớn, với bài toán có không gian state, action nhỏ phương pháp này không đạt hiệu quả cao.

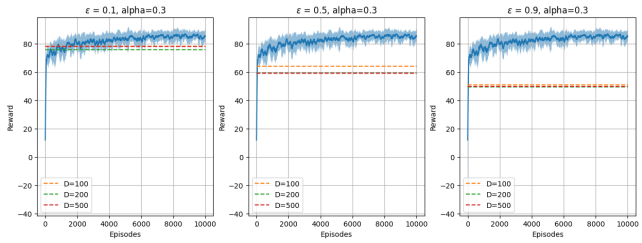

Figure 12: Kết quả của Mô phỏng 6. Ảnh hưởng có mức khám phá ϵ tới phần thưởng của agent.

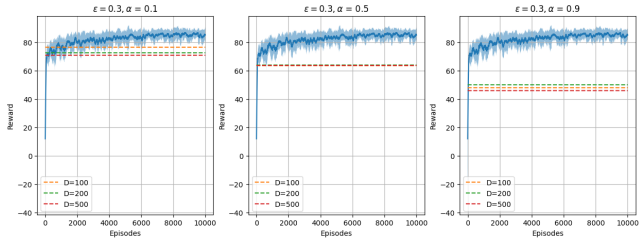

Figure 13: Kết quả của Mô phỏng 6. Ảnh hưởng của tốc độ học α tới phần thưởng của agent.

6 THẢO LUẬN KẾT QUẢ

Các mô phỏng đã cho thấy rằng việc sử dụng học tăng cường (RL) để giải các bài toán Capture the Flag (CTF) là khả thi. Trong quá trình này, vai trò và những thách thức trong việc khám phá cấu trúc bài toán Pentesting (PT) và cung cấp kiến thức cho agent đã được làm rõ.

Chúng tôi đã xem xét hai cách mà cấu trúc của bài toán có thể thay đổi và đặt ra những thách thức cụ thể:

- Cấu trúc vấn đề ngày càng không xác định được, chuyển từ hệ thống CTF cố định sang hệ thống max-entropy, điều này dẫn đến giới hạn của việc học bằng suy luận.
- Một bài toán CTF cố định với cấu trúc ngày càng phức tạp hơn đòi hỏi số lượng mẫu theo cấp số nhân để tác nhân tìm ra cấu trúc của bài toán.

Để giải quyết những thách thức này, việc cung cấp kiến thức ban đầu cho agent là cần thiết.

Các giải pháp cung cấp kiến thức ban đầu cho agent bao gồm:

- **Lazy loading**: Giả định rằng một số cấu hình nhất định trong không gian vấn đề sẽ không bao giờ gặp phải và do đó có thể bị bỏ qua.
- **State aggregation**: Giả định rằng các cấu hình nhất định sẽ giống hệt về mặt thực tế với các cấu hình khác.
- **Imitation learning**: Giả định rằng một giải pháp tối ưu sẽ không quá xa so với những minh chứng nổi tiếng.

Tất cả các dạng kiến thức trước này không bị ràng buộc về mặt ngữ nghĩa với một vấn đề cụ thể và chúng có thể được triển khai trên nhiều vấn đề CTF khác nhau.

Khám phá cấu trúc là một bước thiết yếu trong CTF. Do đó, một agent học lý tưởng cần cân bằng hợp lý giữa việc khám phá cấu trúc và dựa vào kiến thức tiên nghiệm.

Cân nhắc về độ phức tạp và những hạn chế thực tế cho thấy rằng việc đưa ra kiến thức tiên nghiệm có thể là điều cần thiết. Agent hoàn toàn dựa trên mô hình có thể không lý tưởng vì khó mã hóa và không linh hoạt. Tuy nhiên, agent kết hợp thuật toán không có mô hình với kiến thức tiên nghiệm có thể đạt được sự cân bằng lý tưởng để làm cho chúng hiệu quả và hữu ích.

7 HƯỚNG NGHIÊN CỨU TƯƠNG LAI

Các hướng nghiên cứu tương lai được đề xuất bao gồm việc cải tiến các thuật toán học tăng cường để xử lý tốt hơn các môi trường CTF phức tạp hơn. Điều này có thể bao gồm việc áp dụng các kỹ thuật học từ mô phỏng, học từ biểu diễn và các phương pháp học sâu để cải thiện khả năng tổng quát hóa và hiệu quả học tập của các agent. Ngoài ra, cần nghiên cứu thêm về việc tích hợp các phương pháp học tăng cường với các hệ thống bảo mật thực tế để đánh giá hiệu quả và khả năng ứng dụng của chúng.

Mở rộng bài toán CTF, cải thiện cách thức quản lý cấu trúc và kiến thức tiên nghiệm có thể được thực hiện qua nhiều cách:

- Về mặt mở rộng: Tăng quy mô không gian state và action, cũng như tính không tĩnh (non-stationarity) của môi trường.
- Về mặt cấu trúc và tích hợp kiến thức:
  - Chuyển từ các thuật toán bảng (tabular algorithms) sang các thuật toán xấp xỉ (approximate algorithms), hy sinh khả năng diễn giải để đạt được hiệu suất cao hơn.
  - Xem xét khả năng học tập qua nhiều kênh hoặc dựa vào các dạng kiến thức sẵn có khác. Hướng đi đầy hứa hẹn bao gồm: tích hợp lập kế hoạch (planning), phân rã thứ bậc của bài toán CTF thành các nhiệm vụ con (hierarchical decomposition of a CTF in sub-tasks), dựa vào các thiên kiến quy nạp quan hệ (relational inductive biases), hoặc tích hợp kiến thức logic vào quá trình học tập.

Sử dụng phương pháp học mô hình (model learning) cho phép xấp xỉ môi trường và học offline để cải thiện hiệu quả. Điều chỉnh cách cho điểm để hướng dẫn agent học tốt hơn cũng là một hướng nghiên cứu quan trọng.

Cuối cùng, việc áp dụng học chuyển giao (transfer learning) sẽ giúp tận dụng kiến thức đã học từ các bài toán hoặc môi trường khác để cải thiện hiệu quả và tốc độ học tập của agent trong các bài toán CTF mới.

8 KẾT LUẬN

Nghiên cứu này đã chỉ ra tiềm năng của học tăng cường trong việc giải quyết các vấn đề CTF và PT, đồng thời cũng nhấn mạnh những thách thức cần phải vượt qua. Việc sử dụng Q-learning đã cho thấy những kết quả khả quan trong các môi trường đơn giản, nhưng cần phải cải tiến hơn nữa để áp dụng hiệu quả trong các môi trường phức tạp. Những hướng nghiên cứu tương lai và các xem xét đạo đức được đề xuất sẽ đóng vai trò quan trọng trong việc định hình và phát triển lĩnh vực này trong tương lai.

A PHÂN CÔNG NHIỆM VỤ

Table 2: Bảng phân công nhiệm vụ

Nội dung	Phân công	Mức độ hoàn thành
Tìm hiểu lý thuyết	Cả nhóm	100%
Proposal	Huỳnh Nguyễn Uyển Nhi	100%
Thử nghiệm 6 kịch bản	Cả nhóm	100%
Slide	Cả nhóm	100%
Thuyết trình	Trần Minh Duy Phạm Ngọc Thiện	100%
Làm Poster	Trần Minh Duy	100%
Final writeup	Cả nhóm	100%

B TÓM TẮT MỘT SỐ BÀI BÁO CÓ CHỦ ĐỀ TƯƠNG TỰ

K. Pozdniakov, E. Alonso, V. Stankovic, K. Tam, and K. Jones, "Smart computer security audit: Reinforcement learning with a deep neural network approximator," 2020.

Ý tưởng: phát triển AgentPen loại bỏ yếu tố con người khỏi giai đoạn PT và tự động khám phá chiến lược, điều chỉnh và cải thiện logic tấn công trong thời gian thực — cách tiếp cận này không loại bỏ hoàn toàn nhu cầu về chuyên môn về an ninh mạng mà thay vào đó nâng nhà phân tích lên mức độ kiểm soát cao hơn. Các quyết định trên con người vẫn sẽ thúc đẩy:

(1) Việc kiểm tra với các sửa đổi thuật toán học tập cấp cao,

(2) Đạo đức hack, các mối quan tâm và kết quả pháp lý,

(3) Phát triển các chiến lược và giải pháp học máy để tăng hiệu suất.

AgentPen có thể được điều chỉnh, đưa ra các nhiệm vụ và mục tiêu cụ thể, thực hiện các nhiệm vụ khó khăn, thường xuyên, nhàm chán và cho phép nhà phân tích tập trung vào các nhiệm vụ kiểm soát cấp cao hơn.

Áp dụng thuật toán học máy không giám sát, Q-learning, với một công cụ xấp xỉ kết hợp kiến trúc mạng thần kinh sâu (the Elman type RNN). Bản thân việc kiểm tra bảo mật được mô hình hóa như một Quy trình Quyết định Markov để kiểm tra một số chiến lược ra quyết định và so sánh sự hội tụ của chúng với mức tối ưu.

Sử dụng mạng nơ-ron autoencoder để giảm chiều dữ liệu trước khi học.

Một số điểm khác biệt cơ bản với nghiên cứu hiện tại:

- Môi trường máy ảo - gần giống với thực tế
- Sử dụng cả 2 loại biểu diễn Q-table và Q-neural

TABLE I
AUDIT MDP STATES.

Audit MDP state	Description
Init	Initial audit state
Win	The host has a Windows OS
Lx	The host has a Linux OS
P445	The host has an open port 445
P135	The host has an open port 135
P22	The host has an open port 22
P2525	The host has an open port 2525
Serv1	The host has a vulnerable $RDP_{Service}$
Serv2	The host has a vulnerable $Msvcrt_{Service}$
Serv_SSH	The host has a $SSH_{Service}$
$Expl_1$	Remote $Exploit_1$ is executed
$Expl_2$	Remote $Exploit_2$ is executed
$Expl_3$	Local $Exploit_3$ is executed
SSH_brf	SSH brute force attack is performed
$Expl_{12}$	Remote $Exploit_1$ executed as 2^{nd} exploit
$Expl_{22}$	Remote $Exploit_2$ executed as 2^{nd} exploit
$Expl_{32}$	Local $Exploit_3$ executed as 2^{nd} exploit
Fail	Final(terminal) state, dead end, i.e. exploit failed
Succ	Final(terminal) state, target host hacked

Figure 14: Không gian state

TABLE II
AUDIT MDP ACTIONS.

Audit MDP action	Description
Win_check	Checks for Windows OS
Linux_check	Checks for Linux OS
Port445_check	Checks if Port445 is open
Port135_check	Checks if Port135 is open
Port22_check	Checks if Port22 is open
Port2525_check	Checks if Port2525 is open
$Service_1$_check	Check $Service_1$ process is running
$Service_2$_check	Check $Service_2$ process is running
SSH_check	Check SSH process is running
$Execute\_exploit_1$	Grants remote attacker admin privileges
$Execute\_exploit_2$	Grants remote attacker user-level privileges
$Execute_exploit_3$_local	Exploit local services, privilege escalation
Execute_SSH_brf	SSH brute force attack

Figure 15: Không gian actions

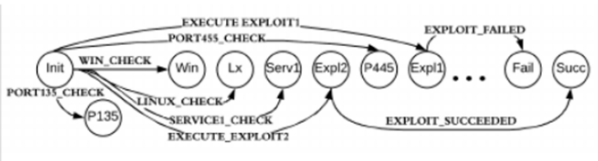

Fig. 2. Examples of MDP audit system transitions from initial state.

Figure 16: Hàm chuyển đổi giữa các state khác nhau

Algorithm 1: Q-Learning Algorithm

```
Initialize Q(state, action);
Observe current state;
for every time step do
    Select action from state;
    if state trigger successful then
        take action;
        check next state' and reward;
        update Q-values (see Eq 2);
    else
        end action;
    end
    state ← state'
end
```

Figure 17: Thuật toán Q-learning

$$Q(s,a) \leftarrow Q(s,a) + \alpha([r + \gamma maxQ(s',a')] - Q(s,a)) \quad (2)$$

Figure 18: Công thức cập nhật hàm Q

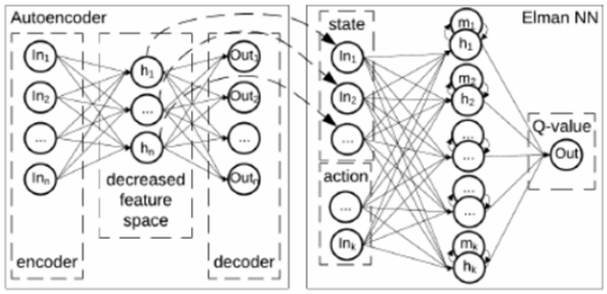

Fig. 3. Deep architecture structure.

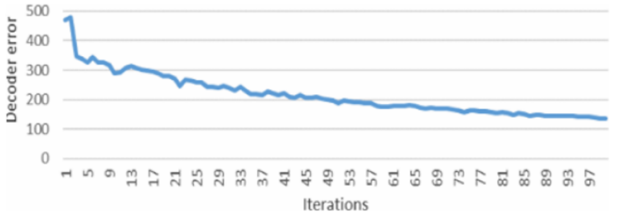

Fig. 6. Autoencoder learning process after 100 episodes/iterations

Fig. 7. RNN approximated Q-Learning performance